Contact detection between convex polyhedra and superquadrics in discrete element codes

Contact detection between convex polyhedra and superquadrics in discrete element codes

Di Peng, Kevin J. Hanley*

School of Engineering, Institute for Infrastructure and Environment, The University of Edinburgh, Edinburgh EH9 3FG, United Kingdom

Abstract

Particle shape substantially influences the bulk behaviour of granular systems. Hence, much scientific interest has been devoted to the adoption of non-spherical fundamental particles in discrete element method simulations. Two examples of such particles are polyhedra, which are highly angular, and superquadrics, which are best suited to simulate rounded particles. It is desirable to use both types of particle together in a simulation to capture the broadest possible range of particle shapes. In this paper, a novel contact detection algorithm is presented for a convex polyhedron and superquadric. This algorithm was implemented in a C++ code which was used to verify the correctness of the algorithm and evaluate its efficiency using the Monte Carlo method. The proposed contact detection algorithm is particularly efficient for many-faceted polyhedra as the effect of increasing the number of faces on the evaluation time is small.

Keywords: discrete element method (DEM), nonspherical particle, polyhedron, superquadric, numerical simulation

1. Introduction

Since its formulation by Cundall and Strack [1] in the 1970s, the discrete element method (DEM) has become extremely popular as it allows exploration of the micro-scale behaviour that determines the bulk-scale response of any particulate system. The most commonly used

*Corresponding author.
Email addresses: di.peng@ed.ac.uk (Di Peng), k.hanley@ed.ac.uk (Kevin J. Hanley)

particle shape in DEM simulations is the sphere, which is both conceptually and computationally simple. The large majority of commercial codes such as PFC [2] and EDEM [3], along with open-source codes such as YADE [4] and LIGGGHTS [5], adopt spheres as the basic shape of element. However, in reality many particles are poorly described by spheres and some problems cannot be adequately addressed without considering particle shape more accurately.

In recognition of the importance of particle shape, there has been a major increase in scientific interest in recent years in DEM modelling of non-spherical particle systems [6]. Many approaches have been proposed. The simplest is implementation of a rolling resistance model at the inter-particle contacts. As particle non-sphericity restricts its rotation to some extent, rolling resistance is capable of capturing a subset of the physics of non-spherical particles but some behaviour is poorly captured. The most common approach for simulating non-spherical particles, adopted in the commercial codes PFC [2] and EDEM [3], is the multi-sphere method in which spheres are clumped together to create irregularly shaped clusters, e.g., [7]. However, many spheres may be required to obtain a reasonable approximation of a real particle's shape [8, 9]. An alternative is to replace the fundamental spheres with other shapes such as polyhedra, ellipsoids [10, 11], or superquadrics [12] which are a generalisation of ellipsoids.

Compared to spheres, accurately resolving the contacts between complex fundamental particles such as superquadrics or polyhedra is difficult and computationally expensive. Two approaches are commonly used: discrete function representation (DFR) and continuous function representation (CFR) [13]. DFR [14, 15] is based on discretising the surface of a particle into a set of nodes. CFR detects and evaluates interparticle contacts through direct manipulation of the equations which mathematically describe a particle's shape. CFR has two subtypes: analytical CFR is used to resolve contacts between polyhedra [16, 17] while numerical CFR is used for resolving contacts between particles with curved surfaces [10, 12, 18, 19, 20]. These CFR approaches tend to be applicable to particles only of specific shapes, e.g., cylinders [21] or convex particles represented using the potential particle approach [22, 23]. There are established analytical CFR methods for detecting and evaluating contacts between poly-

hedra, e.g., the 3DEC [24] code adopts Cundall's contact plane (CP) method [25]. PFC v6.0 allows the simulation of systems of mixed spheres and polyhedra using a variant of the contact detection algorithm proposed by Nezami et al. [26]. Overlaps between ellipsoids may be detected efficiently using CFR-based algorithms, e.g., [27, 28]. A CFR approach for contact resolution of convex superquadrics [29] has been implemented in LIGGGHTS [5].

There are also methods which, in principle, can be applied to any shape of particle. Feng et al. [30] proposed a method of determining contact orientation based on the potential field which is related to particle shape. Dong et al. [31] proposed a contact resolution method based on orientation discretisation: the contact region is divided into cells and each cell is categorised as inside or outside of the particle intersection. Their specific approach requires a pre-computed database. The Gilbert-Johnson-Keerthi (GJK) algorithm [32] is used in computer graphics for detecting penetration or finding the minimum distance between two objects. The original GJK algorithm has been optimised both for speed [33, 34, 35] and for specific shapes such as polyhedra [36]. However, the GJK algorithm is accurate only for shapes with flat surfaces; for shapes with curved surfaces such as superquadrics, discretisation is necessary.

While there are established approaches for polyhedron–polyhedron (PH–PH) and superquadric–superquadric (SQ–SQ) contact detection, an open problem is contact detection in mixed polyhedron–superquadric (PH–SQ) systems. Such simulations are desirable as many real systems contain a broad range of particle shapes. Superquadrics are appropriate for simulating rounded particles but cannot capture the high degree of angularity provided by polyhedra. Hence, having both shapes available in a single simulation gives a broader range of possible particle shapes than solely polyhedra or superquadrics.

In this paper, a novel contact detection algorithm for a convex polyhedron and superquadric is presented which is efficient and reasonably simple. An associated contact evaluation method has also been proposed to determine the contact normal and interparticle overlap. The contact detection algorithm has been implemented in a C++ code, enabling verification of the algorithm and quantification of its run time.

2. Development of convex PH–SQ contact detection algorithm

In Sections 2.1 and 2.2, the superquadric (SQ) and polyhedron (PH) particles are respectively introduced and an existing algorithm for SQ–SQ contact detection is presented. The newly developed algorithm for PH–SQ contact detection is then described in Section 2.3 based on this prior knowledge.

2.1. Definition of and contact detection between convex superquadrics

The surface of a SQ particle may be written as a function

$$F(\boldsymbol{X}) = F(x,y,z) = \left(\left|\frac{x}{a}\right|^{n_2} + \left|\frac{y}{b}\right|^{n_2}\right)^{\frac{n_1}{n_2}} + \left|\frac{z}{c}\right|^{n_1} - 1 = 0 \tag{1}$$

where $\boldsymbol{X}$ is a point defined by three coordinates (x, y, z) in 3D space [29]. a, b, c are shape parameters (half-lengths of the particle along the x, y, z axes) and n_1, n_2 are blockiness parameters which define a SQ's shape. $n_1 = n_2 = 2$ corresponds to an ellipsoid; in addition, if $a = b = c$, the ellipsoid becomes a sphere. The LIGGGHTS implementation of SQs requires $n_1, n_2 \geq 2$ [37]. This prohibition of non-convex SQs is to avoid issues with multiple contacts between two SQs.

If $F(\boldsymbol{X}) < 0$, $\boldsymbol{X}$ lies within the particle; if $F(\boldsymbol{X}) > 0$, $\boldsymbol{X}$ is outside of the particle. The special case $F(\boldsymbol{X}) = 0$ means that $\boldsymbol{X}$ lies on the particle's surface. Consider two SQs A and B that are potentially in contact. The surfaces of both particles can be written as $F_A(\boldsymbol{X})$ and $F_B(\boldsymbol{X})$, or simply F_A and F_B. Contact detection between A and B is equivalent to the following constrained optimisation [23]:

$$\text{Minimise } F_A + F_B \text{ while } F_A - F_B = 0 \tag{2}$$

Podlozhnyuk et al. [29] substituted Eq. (1) into Eq. (2), and used the Lagrange multiplier method to obtain a solution. Suppose the Lagrange function

$$L(x, y, z, \lambda) = F_A + F_B + \lambda(F_A - F_B). \tag{3}$$

Thus,

$$\nabla L = \mathbf{0} \Longleftrightarrow \nabla(F_A + F_B) = -\lambda\nabla(F_A - F_B) \Longleftrightarrow (1+\lambda)\nabla F_A + (1-\lambda)\nabla F_B = \mathbf{0} \tag{4}$$

By combining Eq. (4), which provides three equations, with $\frac{\partial L}{\partial \lambda} = 0$, or equivalently $F_A - F_B = 0$, a set of four equations is obtained:

$$\mathbf{\Phi} = \begin{bmatrix} (1+\lambda)\frac{\partial F_A}{\partial x} + (1-\lambda)\frac{\partial F_B}{\partial x} \\ (1+\lambda)\frac{\partial F_A}{\partial y} + (1-\lambda)\frac{\partial F_B}{\partial y} \\ (1+\lambda)\frac{\partial F_A}{\partial z} + (1-\lambda)\frac{\partial F_B}{\partial z} \\ F_A - F_B \end{bmatrix} = \mathbf{0} \tag{5}$$

There must be a solution $(x_0, y_0, z_0, \lambda_0)$ to Eq. (5). The multi-unknown Newton's method can be used to obtain this solution:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \\ \lambda_{n+1} \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \\ z_n \\ \lambda_n \end{bmatrix} - \boldsymbol{J}^{-1}\mathbf{\Phi} \tag{6}$$

where $\boldsymbol{J}$ is the Jacobian of $\mathbf{\Phi}$. The detailed solution procedure can be found in Podlozhnyuk et al. [29], resulting in the point (x_0, y_0, z_0), which is the midpoint of two points $\boldsymbol{X_A}$ and $\boldsymbol{X_B}$: the closest point to the centre of SQ B on SQ A's surface and the closest point to the centre of SQ A on SQ B's surface, respectively. We can evaluate whether or not (x_0, y_0, z_0) is the contact point with the following criterion:

$$F_A(x_0, y_0, z_0) < 0 \quad \text{and} \quad F_B(x_0, y_0, z_0) < 0 \tag{7}$$

If Eq. (7) is true, A and B are in contact; otherwise the two SQs are not in contact. If F_B were substituted by the function of a planar wall, SQ–wall contact detection can be accomplished in a similar manner to find the point on the surface of the SQ which is closest

to the wall, or the deepest point beneath the wall surface if the SQ contacts the wall [29].

2.2. Definition of and points within convex polyhedra

The surface function describing a polyhedron is given as Eq. (8) [22]:

$$F(\boldsymbol{X}) = F(x, y, z) = \sum_{i=1}^{n} \langle f_i(x, y, z) \rangle = \sum_{i=1}^{n} \langle A_i x + B_i y + C_i z - D_i \rangle = 0 \tag{8}$$

where n is the total number of faces comprising the PH, $f_i(x, y, z)$ is the function of the i^{th} plane of the PH with outer normal vector (A_i, B_i, C_i), and D_i is the perpendicular distance from the origin to the i^{th} plane of the PH. The Macaulay brackets in Eq. (8) indicate that positive terms retain their values while negative terms are set to zero. Eq. (8) is equivalent to

$$F(\boldsymbol{X}) \leq 0 \Longleftrightarrow \forall f_i(\boldsymbol{X}) \leq 0 \tag{9a}$$

$$F(\boldsymbol{X}) > 0 \Longleftrightarrow \exists f_i(\boldsymbol{X}) > 0. \tag{9b}$$

Eq. (9a) means that if a point is within a PH, all PH surface functions must be negative at that point. Conversely, if a point makes one or more PH surface functions positive, that point must be outside the PH (Eq. (9b)). Thus Eq. (9a) can be used to confirm a possible contact point, while Eq. (9b) can be used to eliminate invalid candidate contact points. Eq. (9) is used extensively in the algorithm presented in Section 2.3.

2.3. Resolving contact or non-contact between a PH and a SQ

In total, there are six distinct cases that must be considered in this algorithm: three cases in which the PH and SQ are in contact, and three in which they are not. We have chosen a triangular prism with five faces as the PH and an ellipsoid as the SQ for most of the figures in Section 2.3. After the six cases have been described, a flowchart is shown as Fig. 7 which summarises the steps in the algorithm.

2.3.1. Defining the 'Initial Face' (IF) and positive/negative faces

We define the initial face (IF) as the face of the PH whose outer normal vector has the maximum dot product, i.e., is most closely aligned, with the vector joining the centroids of the PH and SQ. An example IF is shown in Fig. 1(a). The IF is used in the first step to judge whether or not the particles are in contact. Positive/negative faces are assigned that status by their positions relative to a fixed point $\boldsymbol{X}$. Each face lies on an infinite plane; if the point $\boldsymbol{X}$ is above that plane, where the outer normal vector indicates the positive direction, that face is positive for point $\boldsymbol{X}$. Conversely, if $\boldsymbol{X}$ is beneath the plane, the face is negative for $\boldsymbol{X}$. In order to clearly present a 3D shape on paper, only three of the faces of the triangular prism PH are labelled as positive or negative in the figures in this section; the other faces are always negative.

In this algorithm, the SQ–planar wall contact detection approach summarised in Section 2.1 [29] is applied to one or more specific planes enclosing faces of the PH. This yields one or more points on the surface of the SQ which are potentially contact points.

2.3.2. Case 1: Positive IF

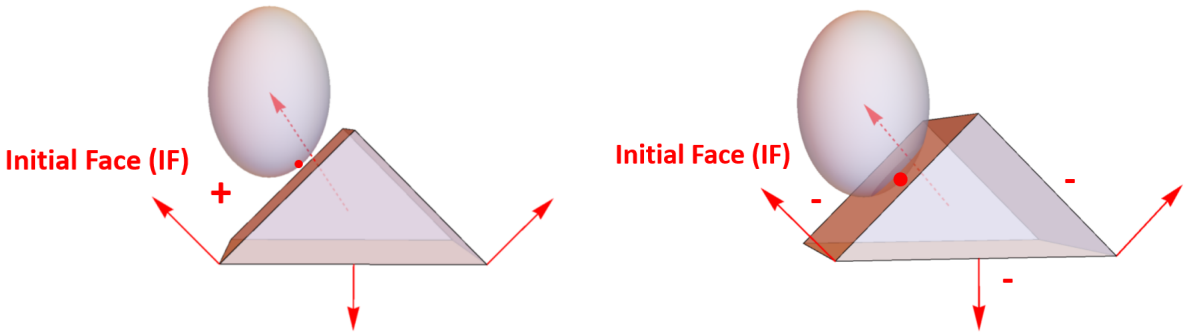
Consider Fig. 1(a) in which the PH and SQ are not in contact. The point highlighted on the surface of the SQ has been identified as a potential contact point based on Section 2.1. However, this point makes the IF positive. Based on Eq. (9b), there must be no contact and the contact detection can be terminated. Case 1 is highly efficient as there is no need to check any PH faces apart from the IF.

2.3.3. Case 2: All faces negative

Case 2, in which all faces are negative, is shown in Fig. 1(b). The highlighted point must be the contact point according to Eq. (9a). Case 2 is somewhat less efficient than Case 1 as the positive/negative assessment must be made for all faces of the PH, not only the IF.

2.3.4. Cases 3 & 4: Negative IF, resolve through switch to 'Positive Face' (PF)

Cases 3 (non-contact) and 4 (contact) arise when the IF is negative for the potential contact point denoted as $\boldsymbol{X}_0$ but one other face is positive. The scenario in which more



(a) Case 1: non-contact due to a positive IF

(b) Case 2: the simplest contact case with all PH faces negative

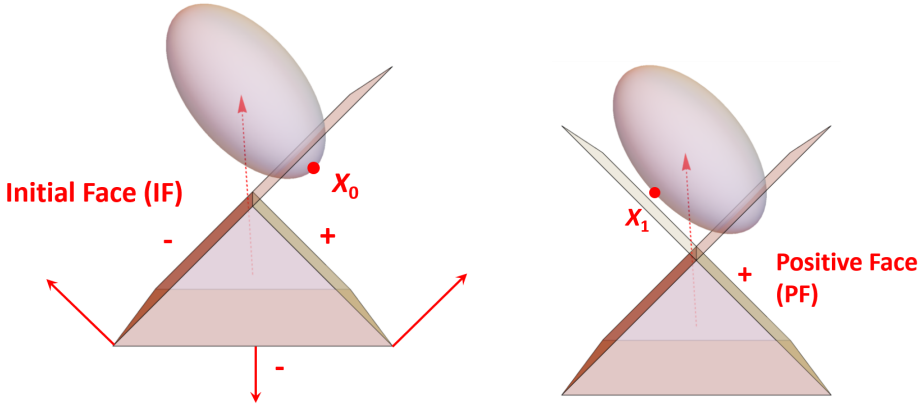
Figure 1: The simplest (non-)contact cases for face contact of a PH and SQ.

than one face is positive is addressed in Section 2.3.5. Fig. 2(a) illustrates Case 3: $\boldsymbol{X}_0$ makes one face, which is not the IF, positive. In this situation, we switch from the IF to this newly identified positive face (PF) and repeat the same procedure as before. A new potential contact point $\boldsymbol{X}_1$ is identified by applying the procedure outlined in Section 2.2 to the plane which includes the PF (Fig. 2(b)). Since the PF remains positive, the same conclusion of no contact is drawn, based on Eq. (9b), as for Case 1.

Case 4 is the analogue of Case 3 for which there is a PH–SQ contact. This is shown in Fig. 3. As usual, the IF is found initially, resulting in a potential contact point $\boldsymbol{X}_0$ (Fig. 3(a)). Since $\boldsymbol{X}_0$ makes a face other than the IF positive, it cannot be the contact point but another point on the SQ surface may still be in contact with the PH. Attention is switched from the IF to the PF (Fig. 3(b)). The potential contact point $\boldsymbol{X}_1$ makes all PH faces negative. As concluded for Case 2, $\boldsymbol{X}_1$ is the contact point.

2.3.5. Cases 5 & 6: Negative IF, search and return for edge/vertex contact

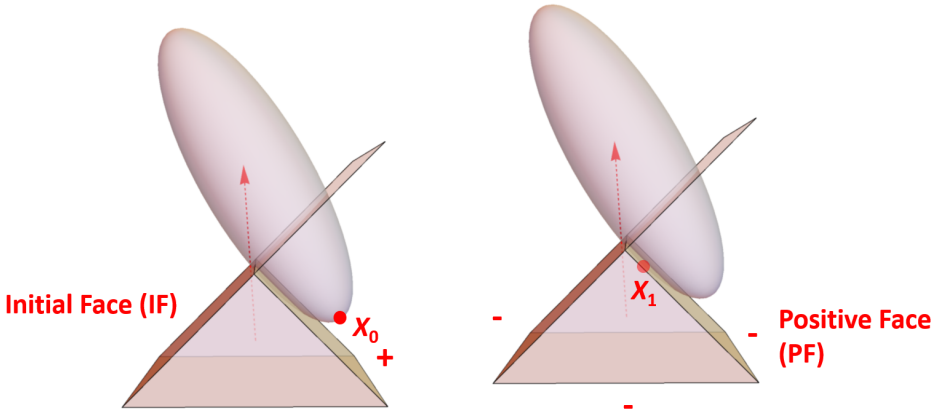
In Cases 3 and 4, an unambiguous assessment of contact/non-contact could be made after switching from the initial face to the one positive face. However, one could imagine a scenario in which the contact status remains uncertain after switching to the one PF. For example, resolving a contact between a SQ and the edge or vertex of a PH, rather than its face, cannot be adequately resolved using Cases 1–4. Furthermore, Cases 3 and 4 are



(a) Positive and negative faces and potential contact point $\boldsymbol{X}_0$ based on the IF

(b) After switching attention to the PF

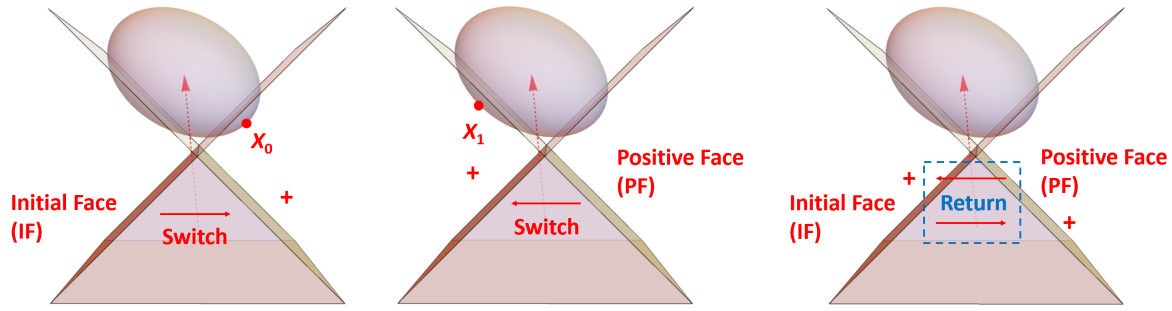
Figure 2: Non-contact of a PH and SQ determined by switching from the IF to the PF (Case 3).



(a) Positive and negative faces and potential contact point $\boldsymbol{X}_0$ based on the IF

(b) Contact point $\boldsymbol{X}_1$ found after switching to the PF

Figure 3: Contact point $\boldsymbol{X}_1$ found after switching from the IF to the PF (Case 4).



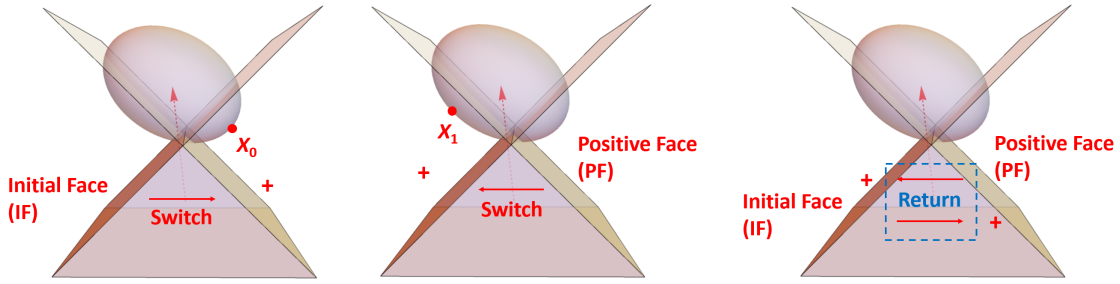
(a) First switch from IF to PF (b) Second switch from PF back to IF (c) Check the common edge

Figure 4: SR method for determining SQ contact with a PH edge/vertex (Case 5: non-contact).

inapplicable if there is more than one positive face. In these situations, Cases 5 (non-contact) and 6 (contact) apply.

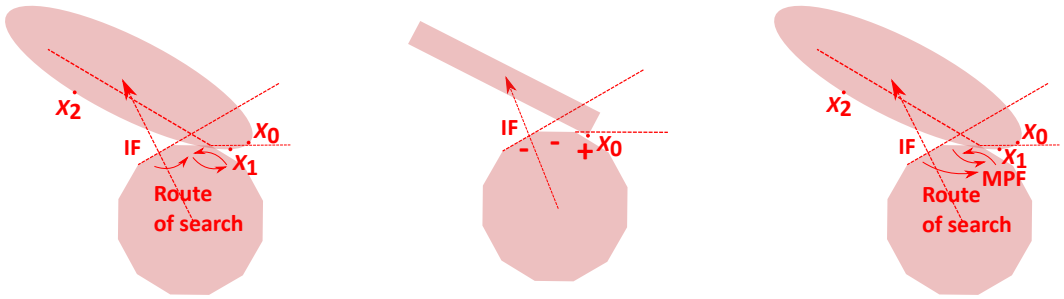
Consider Fig. 4(a). As for Cases 3 and 4, a potential contact point $\boldsymbol{X}_0$ is found initially using the IF obtained by joining the centroids of the two shapes. Since this makes the rightmost face positive, another potential contact point $\boldsymbol{X}_1$ is found based on the PF (Fig. 4(b)). However, $\boldsymbol{X}_1$ makes the PF negative but the IF positive, thus creating an endless alternating cycle between these two faces. This is resolved by checking whether the edge common to the two faces contacts the SQ (Fig. 4(c)). Fig. 5 is a similar scenario in which there is indeed a contact with the common edge. This is termed the 'search and return' (SR) method for contact determination. This also allows the detection of contacts between a SQ and a vertex of a PH: since a vertex belongs to multiple edges of a PH, contact can be assessed using any of the edges linking to it.

In Figs. 4 and 5, the return appears after only one switch from IF to another face. However, more iterations may be needed before a return appears, e.g., Fig. 6(a). This is a limitation of the simple SR method. Furthermore, consider Fig. 6(b) which shows a 2D projection of a cuboidal SQ contacting a many-faceted PH. Starting from the IF in the usual manner, both the IF and all of the faces surrounding it are negative. However, since one face of the PH remains positive, $\boldsymbol{X}_0$ cannot be definitively identified as the contact point. The SR method described above needs to be amended to account for this general situation.



(a) First switch from IF to PF (b) Second switch from PF back to IF (c) Check the common edge

Figure 5: SR method for determining SQ contact with a PH edge/vertex (Case 6: contact found).



(a) Multiple searches before a return (b) Positive face not adjacent to IF (c) New search route for the amended SR method

Figure 6: Extreme PH–SQ contact situations which cannot be addressed using the simple SR method.

This is simply done. As before, start from the IF and find $\boldsymbol{X}_0$. The check of plane function values with $\boldsymbol{X}_0$ is expanded to all PH faces, rather than those faces adjacent to the IF. The face with maximum positive value (termed the 'maximal positive face' or MPF) becomes the starting point for the next search. This amended initial search procedure is suitable for Fig. 6(a), as demonstrated in Fig. 6(c), and is also applicable to all other scenarios, including Fig. 6(b). As the SQ surface is considered in finding the MPF, this amended SR method also improves efficiency by reducing the scope of the search to the area in which contact is most likely. Only neighbouring faces need to be checked in the following searches to find a return.

There is a subtle difference to note between contacts with the faces of PHs, e.g., Fig. 1(b) or Fig. 3, and contacts with the edges or vertices of PHs, e.g., Fig. 5 or Fig. 6. Face contacts yield a contact point on the surface of the SQ while edge/vertex contacts yield a contact point which is on an edge of the PH.

This algorithm is only applicable to a convex PH and SQ: the outer normal vector is used in the approach for PHs, while the SQs must be convex to avoid the possibility of multiple contacts as the algorithm presented in this section gives at most one contact point. Furthermore, the overlap between a PH and a SQ should be small to ensure accuracy. This is because an identified contact point is always located on the surface of the PH or SQ; as overlaps become large, the validity of choosing a surface point as the single contact point becomes questionable.

The complete algorithm is summarised in Fig. 7. The six cases leading to termination of the algorithm are shown in this flowchart. For odd cases, a contact is not found; for even cases, a contact is found.

2.4. Contact evaluation based on the contact point

If a contact exists, a single contact point $\boldsymbol{X}_c$ is found by following the algorithm in Fig. 7. The next step is to determine the contact normal and the interparticle overlap, both of which are necessary to calculate a contact force. It is noted that these quantities are not well defined in general for irregularly shaped particles [21]; however, the approach proposed

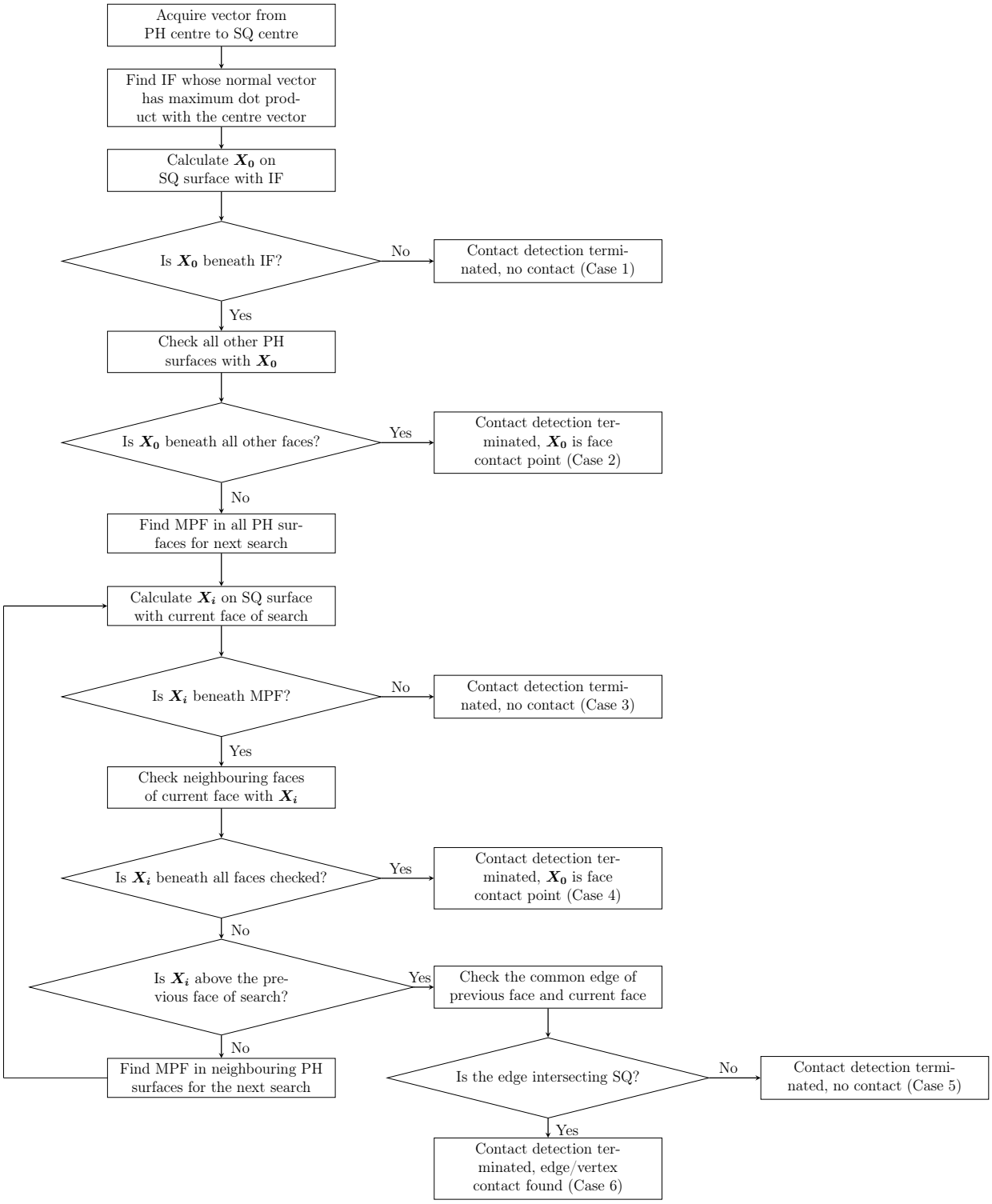


Figure 7: Flowchart showing the complete PH–SQ contact detection algorithm.

here gives physically reasonable values. Three cases must be considered separately: face contact, edge contact and vertex contact.

For face contact, the contact point lies on a single face of the PH which is therefore the contact plane. The normal vector of the contact plane is

$$\boldsymbol{n} = \boldsymbol{n}_c \tag{10}$$

where $\boldsymbol{n}_c$ is the normal vector of the PH face which contains the contact point $\boldsymbol{X}_c$. The interparticle overlap is the distance between $\boldsymbol{X}_c$ and the contact plane (Fig. 8(a)). This method originates from Feng et al. [30].

For edge contact, it would not be physically meaningful to adopt either of the PH faces forming the edge as the contact plane. Hence, a virtual contact plane is sought. Its normal vector is chosen to be the normalised sum of the normal vectors of the two faces forming the edge:

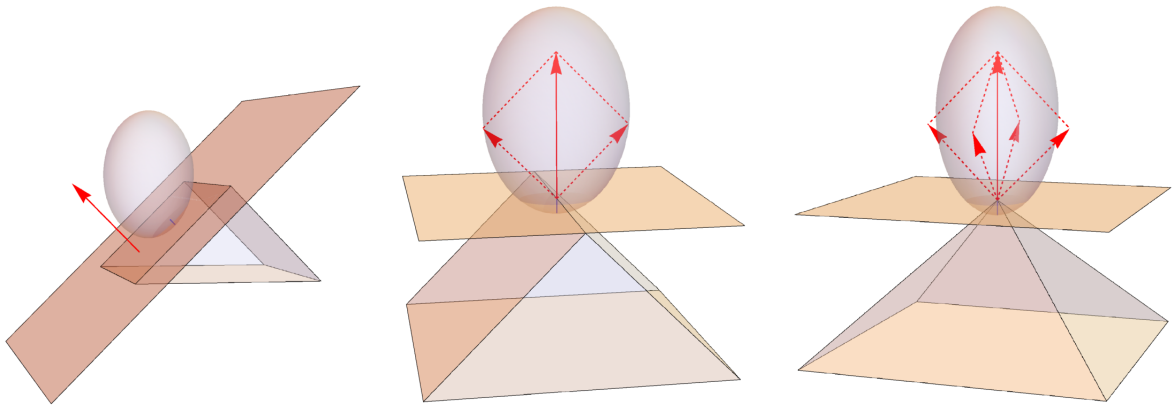
$$\boldsymbol{n} = \frac{\boldsymbol{n}_1 + \boldsymbol{n}_2}{||\boldsymbol{n}_1 + \boldsymbol{n}_2||} \tag{11}$$

where $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ are the normal vectors of the two faces forming the edge. The contact plane passes through $\boldsymbol{X}_c$ with normal vector $\boldsymbol{n}$. The overlap is computed between the SQ and the contact plane [29] (Fig. 8(b)).

For vertex contact, a virtual contact plane is again required. Its normal vector is chosen to be the normalised sum of the direction vectors $\boldsymbol{l}$ of all n edges meeting at the vertex:

$$\boldsymbol{n} = \frac{\sum_{i=1}^{n} \boldsymbol{l}_i}{||\sum_{i=1}^{n} \boldsymbol{l}_i||} \tag{12}$$

$\boldsymbol{l}$ always points towards the vertex contact. This notation emphasises the adoption of the direction vectors of edges rather than normal vectors of faces for the edge contact case. As for the edge contact case, the overlap is computed between the SQ and the contact plane [29] (Fig. 8(c)).



(a) For face contact (b) For edge contact (c) For vertex contact

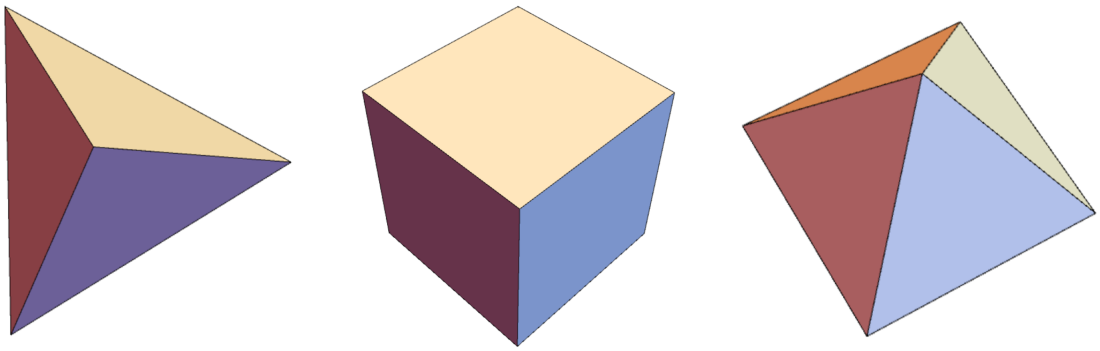
Figure 8: Contact evaluation for three different contact cases.

3. Implementation and verification of the contact detection algorithm

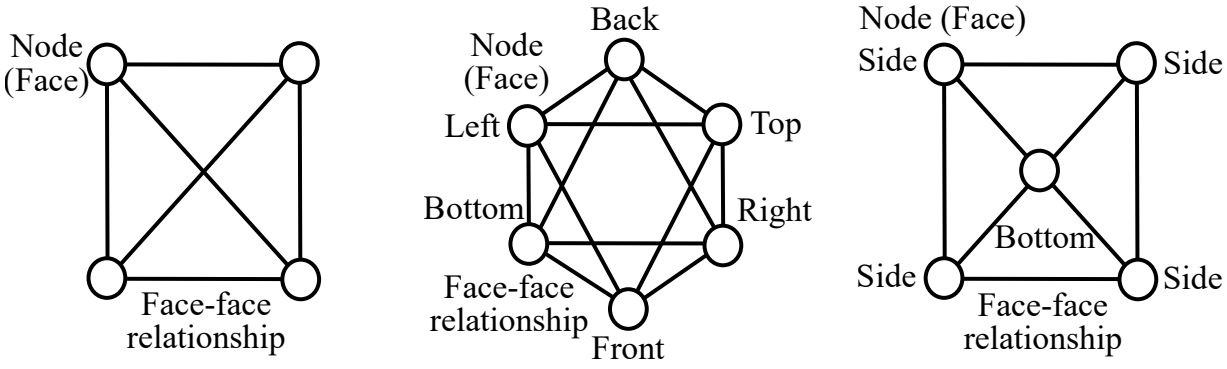
3.1. Implementation of algorithm

LIGGGHTS-PUBLIC v3.8.0 [37] implements the contact detection approach described in Section 2.1 for SQ–SQ, SQ–planar wall and SQ–line segment contacts. The LIGGGHTS implementation was used as a basis for the code implementation described in this paper. Since the algorithm requires knowledge of neighbouring faces, it is beneficial to build a relationship map of PH surface faces into the implementation. A graph model in which nodes are PH surface faces and links are relationships between these faces was adopted to satisfy this requirement. Two faces link only when they have a common edge, not if they only share a vertex. Fig. 9 shows three simple examples of this graph model. All faces are linked to each other for a tetrahedron; for a cube, each face is linked to four other faces; for a regular pyramid, any side face links to all other faces except the face directly opposite, while the square bottom face links to all faces.

This PH model was implemented in C++ by creating a class which includes a set of planes enclosing the PH faces (stored using the outer normal vector and one specific point on each plane) and a 2D adjacency array to store the relationships between faces/planes. This is an easy way to model a static graph [38]. Within this class, the vertices are stored in a disordered array list. The vertices are traversed only when edge/vertex contact judgement is

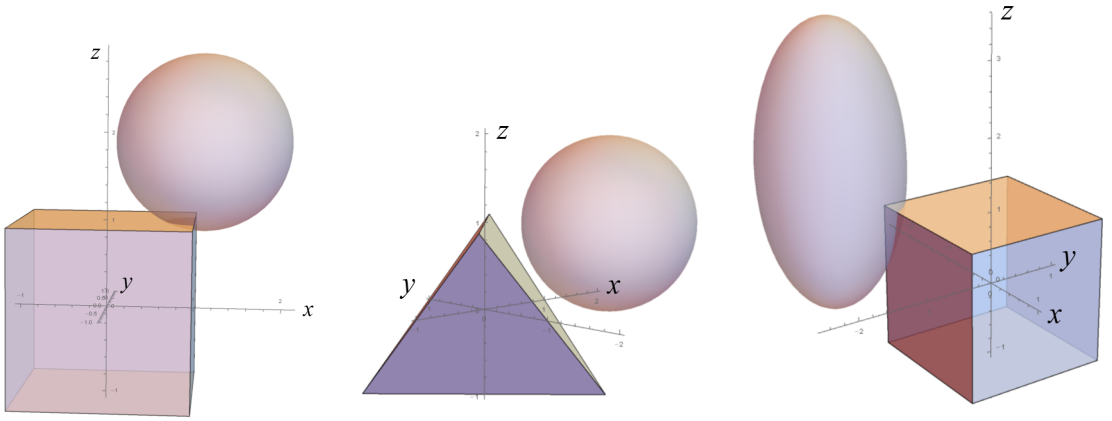


(a) Tetrahedron (b) Cube (c) Regular pyramid



(d) Tetrahedron model (e) Cube model (f) Regular pyramid model

Figure 9: Describing PHs using a graph model.



(a) Sphere–cube edge contact (b) Sphere–tetrahedron non-contact (c) Ellipsoid–cube vertex contact

Figure 10: Three examples of verification cases.

needed. This traversal was not optimised to improve upon the time complexity of $O(n)$ [39] for the naive implementation which was considered acceptable. This class was combined with the SQ implementation from LIGGGHTS to create a standalone program for verifying one PH contacting one SQ.

3.2. Verification of algorithm

Initially, spheres, the simplest SQs, were placed in a wide range of contacting and non-contacting positions relative to a cube, e.g., Fig. 10(a). Contact/non-contact was determined using the code described in Section 3.1. The contact points identified were compared with the analytical contact points to confirm that the correct results were obtained. A wide range of other (non-)contact cases were checked, e.g., sphere–tetrahedron (Fig. 10(b)) and ellipsoid–cube (Fig. 10(c)), which also gave the expected results.

After successfully carrying out this initial stage of verification, the Monte Carlo method was adopted to randomly position a spherical SQ of radius 1 m in a large cubical space (edge length 6 m) containing a smaller cubical PH of edge length 2 m. The centre of the SQ could be any point within the 6 m × 6 m × 6 m cubical space shown in pink in Fig. 11. The cubical PH is shown in green in this figure. The centres of any spheres which contact the cube must lie within the round-cornered cubic space shown in yellow in Fig. 11. 1000000

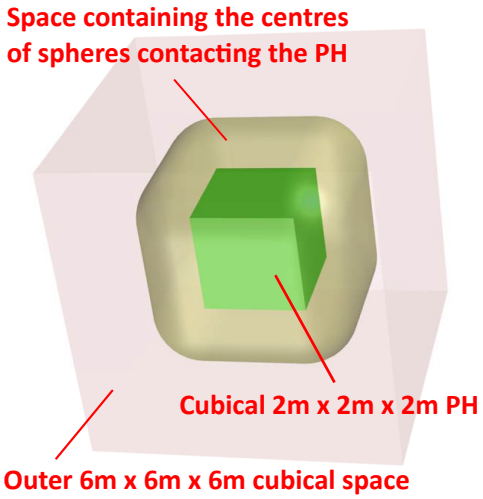


Figure 11: Setup for the Monte Carlo verification.

points were randomly generated within the pink cube as SQ centre points. This led to three possible outcomes for each point, with reference to Fig. 11:

1. The centre was inside the green PH. These points were neither identified as contacts nor as non-contacts but were counted separately as 'Case 8'.
2. The centre was inside the yellow round-cornered cubic space but not inside the green PH. These are contact points, i.e., Cases 2, 4 or 6.
3. The centre was outside the yellow region, i.e., no contact (Cases 1, 3 or 5).

The code described in Section 3.1 was applied to each SQ centre point generated, leading to one of these three outcomes for each point. Analytically, the frequency with which these outcomes were observed must be proportional to the volumes of the three regions on Fig. 11. $V_{green} = 2^3 = 8\,\mathrm{m}^3$, $V_{yellow} \approx 47.038\,\mathrm{m}^3$, and $V_{pink} = 6^3 - V_{green} - V_{yellow} \approx 160.962\,\mathrm{m}^3$. This is exactly what was observed, as shown in Table 1. The difference between the analytical and Monte Carlo results is less than 0.05%, further demonstrating the reliability of this algorithm for resolving PH–SQ contact.

Table 1: Comparison of the Monte Carlo verification with the analytical result based on volumes.

	Cases 1/3/5 (no contact)	Cases 2/4/6 (contact)	Case 8	Total
Monte Carlo count	745619	217499	36882	1000000
Monte Carlo (%)	74.56	21.75	3.69	100.00
Analytical volume (m^3)	160.962	47.038	8	216
Analytical %	74.52	21.78	3.70	100.00

4. Computational efficiency of algorithm

After the code had been verified, it could be used to assess the computational efficiency of the contact detection algorithm. A similar Monte Carlo approach was used for this purpose with the centre points of 10000 spherical SQs being randomly located within the space shown on Fig. 11. Instead of only a cubical PH, all five convex regular polyhedra were used for assessing computational efficiency: a tetrahedron, cube, octahedron, dodecahedron and icosahedron which have 4, 6, 8, 12 and 20 faces, respectively. Sphere centres within the PHs (Case 8) were disregarded for this benchmarking exercise.

Each determination of contact/non-contact was repeated 5000 times to ensure robust data were gathered for total elapsed time. Times were measured using the *ctime* library [40]; this was confirmed to give almost identical results to class *std::chrono::system_clock* [41]. All benchmarks were run on a laptop computer with an Intel Core i7-6600U 2.60 GHz CPU.

The results are shown in Fig. 12. The elapsed calculation times are categorised by case and accumulated curves are adopted for representation. It is noted that, for certain PHs, some cases cannot occur, e.g., Case 4 for a cube. For any PH, Cases 1 and 2 are clearly the fastest, then Cases 3 and 4 where present, and finally Cases 5 and 6 are the slowest by a significant margin, which is expected from Fig. 7. The odd, non-contact cases are generally slightly faster than the corresponding contact cases, as termination of the algorithm occurs sooner for the non-contact cases than for the contact cases.

The effect of increasing the number of PH faces (and also edges and vertices) is perhaps not as pronounced as might be expected. Comparing the graphs for the five different polyhedra on Fig. 12, there is a small time penalty associated with increasing the number of

faces but this is modest. This algorithm is therefore particularly efficient for polyhedra with a large number of faces.

5. Conclusions

This paper presents a novel contact detection algorithm for a convex polyhedron (PH) and superquadric (SQ) in DEM. This algorithm is based on contact detection between a SQ and a planar wall, and terminates in one of six cases for each contact event: three cases in which the PH and the SQ are in contact, and three in which they are not. This contact detection algorithm has been implemented as a standalone program, based on the existing SQ implementation in LIGGGHTS [37].

This program was used to verify the correctness of the algorithm and evaluate its efficiency using the Monte Carlo method. The evaluation time largely depends on the point at which the algorithm terminates. Non-contact cases are generally slightly faster than the equivalent contact cases. The effect of increasing the number of PH faces on the evaluation time is small, indicating that the proposed algorithm is particularly efficient for many-faceted PHs.

Acknowledgements

Dr Andrew Firth from the School of Engineering at The University of Edinburgh is acknowledged. Funding from the University of Edinburgh Global Research Scholarship and the China Scholarship Council (No. 201606260230) is also acknowledged.

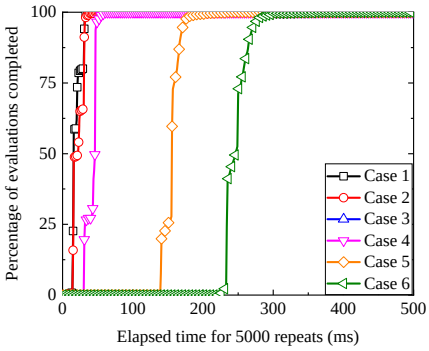
References

[1] P. A. Cundall, O. D. L. Strack, A discrete numerical model for granular assemblies, Géotechnique 29 (1) (1979) 47–65.

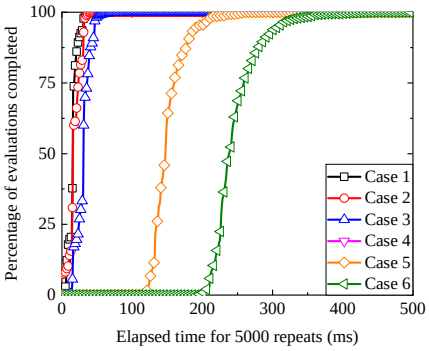
[2] Itasca Consulting Group, Inc., Minneapolis: Itasca, PFC - Particle Flow Code, Ver. 6.0 (2018).

[3] DEM Solutions Ltd., https://www.edemsimulation.com/, EDEM 2019 User Guide (2019).

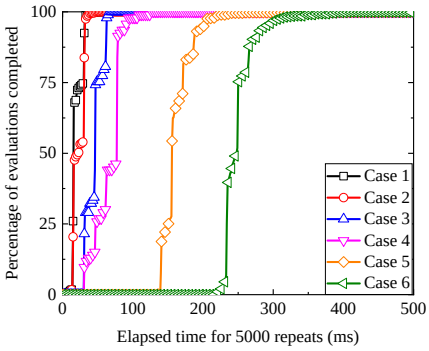
[4] V. Šmilauer et al, Reference manual, in: Yade Documentation 2nd ed, The Yade Project, 2015, http://yade-dem.org/doc/. doi:10.5281/zenodo.34045.



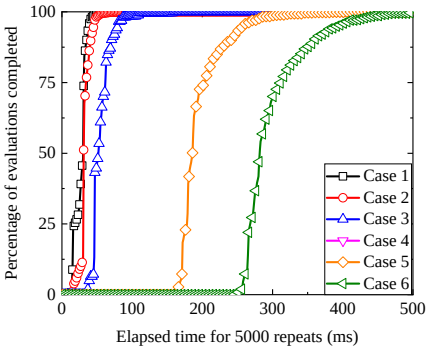
(a) Tetrahedron



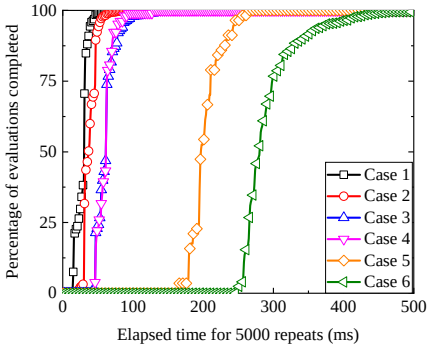
(b) Cube



(c) Octahedron



(d) Dodecahedron



(e) Icosahedron

Figure 12: Evaluation times, subdivided by case, for different types of PH.

[5] C. Kloss, C. Goniva, A. Hager, S. Amberger, S. Pirker, Models, algorithms and validation for opensource DEM and CFD–DEM, Progress in Computational Fluid Dynamics, an International Journal 12 (2–3) (2012) 140–152.

[6] G. Lu, J. R. Third, C. R. Müller, Discrete element models for non-spherical particle systems: from theoretical developments to applications, Chemical Engineering Science 127 (2015) 425–465.

[7] D. Markauskas, R. Kačianauskas, A. Džiugys, R. Navakas, Investigation of adequacy of multi-sphere approximation of elliptical particles for DEM simulations, Granular Matter 12 (1) (2010) 107–123.

[8] H. Kruggel-Emden, S. Rickelt, S. Wirtz, V. Scherer, A study on the validity of the multi-sphere discrete element method, Powder Technology 188 (2) (2008) 153–165.

[9] M. Marigo, E. H. Stitt, Discrete element metod (DEM) for industrial applications: comments on calibration and validation for the modelling of cylindrical pellets, KONA Powder and Particle Journal 32 (2015) 236–252.

[10] X. Lin, T.-T. Ng, Contact detection algorithms for three-dimensional ellipsoids in discrete element modelling, International Journal for Numerical and Analytical Methods in Geomechanics 19 (9) (1995) 653–659.

[11] X. Lin, T.-T. Ng, A three-dimensional discrete element model using arrays of ellipsoids, Géotechnique 47 (2) (1997) 319–329.

[12] J. R. Williams, A. P. Pentland, Superquadrics and modal dynamics for discrete elements in interactive design, Engineering Computations 9 (2) (1992) 115–127.

[13] C. Hogue, Shape representation and contact detection for discrete element simulations of arbitrary geometries, Engineering Computations 15 (3) (1998) 374–390.

[14] J. R. Williams, R. O'Connor, A linear complexity intersection algorithm for discrete element simulation of arbitrary geometries, Engineering Computations 12 (2) (1995) 185–201.

[15] J. R. Williams, R. O'Connor, Discrete element simulation and the contact problem, Archives of Computational Methods in Engineering 6 (4) (1999) 279–304.

[16] E. G. Nezami, Y. M. A. Hashash, D. Zhao, J. Ghaboussi, Simulation of front end loader bucket–soil interaction using discrete element method, International Journal for Numerical and Analytical Methods in Geomechanics 31 (9) (2007) 1147–1162.

[17] Y. T. Feng, D. R. J. Owen, A 2D polygon/polygon contact model: algorithmic aspects, Engineering Computations 21 (2/3/4) (2004) 265–277.

[18] G. Lu, J. R. Third, C. R. Müller, Critical assessment of two approaches for evaluating contacts between super-quadric shaped particles in DEM simulations, Chemical Engineering Science 78 (2012) 226–235.

[19] G. G. W. Mustoe, M. Miyata, Material flow analyses of noncircular-shaped granular media using discrete element methods, Journal of Engineering Mechanics 127 (10) (2001) 1017–1026.

[20] J. E. Andrade, K.-W. Lim, C. F. Avila, I. Vlahinić, Granular element method for computational particle mechanics, Computer Methods in Applied Mechanics and Engineering 241 (2012) 262–274.

[21] Y. Feng, K. Han, D. Owen, A generic contact detection framework for cylindrical particles in discrete element modelling, Computer Methods in Applied Mechanics and Engineering 315 (2017) 632–651.

[22] C. W. Boon, G. T. Houlsby, S. Utili, A new contact detection algorithm for three-dimensional non-spherical particles, Powder Technology 248 (2013) 94–102.

[23] G. T. Houlsby, Potential particles: a method for modelling non-circular particles in DEM, Computers and Geotechnics 36 (6) (2009) 953–959.

[24] Itasca Consulting Group, Inc., Minneapolis: Itasca, 3DEC - Three-Dimensional Distinct Element Code, Ver. 5.2 (2016).

[25] P. A. Cundall, Formulation of a three-dimensional distinct element model. Part I. A scheme to detect and represent contacts in a system composed of many polyhedral blocks, International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts 25 (3) (1988) 107–116.

[26] E. G. Nezami, Y. M. A. Hashash, D. Zhao, J. Ghaboussi, Shortest link method for contact detection in discrete element method, International Journal for Numerical and Analytical Methods in Geomechanics 30 (8) (2006) 783–801.

[27] W. Xu, H. Chen, Mesostructural characterization of particulate composites via a contact detection algorithm of ellipsoidal particles, Powder Technology 221 (2012) 296–305.

[28] Z. Zhu, W. Xu, H. Chen, The fraction of overlapping interphase around 2D and 3D polydisperse non-spherical particles: Theoretical and numerical models, Computer Methods in Applied Mechanics and Engineering 345 (2019) 728–747.

[29] A. Podlozhnyuk, S. Pirker, C. Kloss, Efficient implementation of superquadric particles in discrete element method within an open-source framework, Computational Particle Mechanics 4 (1) (2017) 101–118.

[30] Y. Feng, K. Han, D. Owen, Energy-conserving contact interaction models for arbitrarily shaped discrete elements, Computer Methods in Applied Mechanics and Engineering 205 (2012) 169–177.

[31] K. Dong, C. Wang, A. Yu, A novel method based on orientation discretization for discrete element modeling of non-spherical particles, Chemical Engineering Science 126 (2015) 500 – 516.

[32] E. G. Gilbert, D. W. Johnson, S. S. Keerthi, A fast procedure for computing the distance between complex objects in three-dimensional space, IEEE Journal on Robotics and Automation 4 (2) (1988) 193–203.

[33] C. J. Ong, E. G. Gilbert, The Gilbert-Johnson-Keerthi distance algorithm: A fast version for incremental motions, in: Proceedings of International Conference on Robotics and Automation, Vol. 2, IEEE, 1997, pp. 1183–1189.

[34] G. V. den Bergen, A fast and robust GJK implementation for collision detection of convex objects, Journal of Graphics Tools 4 (2) (1999) 7–25.

[35] S. Cameron, A comparison of two fast algorithms for computing the distance between convex polyhedra, IEEE transactions on Robotics and Automation 13 (6) (1997) 915–920.

[36] S. Cameron, Enhancing GJK: Computing minimum and penetration distances between convex polyhedra, in: Proceedings of international conference on robotics and automation, Vol. 4, IEEE, 1997, pp. 3112–3117.

[37] DCS Computing GmbH, https://www.cfdem.com/media/DEM/docu/Manual.html, LIGGGHTS(R)-PUBLIC Documentation, version 3.x Edition (2016).

[38] K. Mehlhorn, Algorithms and Data Structures: The Basic Toolbox, Springer, Berlin, 2008.

[39] R. Sedgewick, Algorithms, Addison-Wesley Series in Computer Science, Addison-Wesley, Reading, Mass., 1984.

[40] International Organization for Standardization, Programming languages – C (ISO/IEC 9899:1990) (1990).

[41] International Organization for Standardization, Information technology – Programming languages – C++ (ISO/IEC 14882:2011) (2011).